Efficient Algorithms for Weighted Rank-Maximal Matchings and Related Problems

Telikepalli Kavitha Chintan D. Shah

{kavitha,chintan}@csa.iisc.ernet.in
Indian Institute of Science, Bangalore

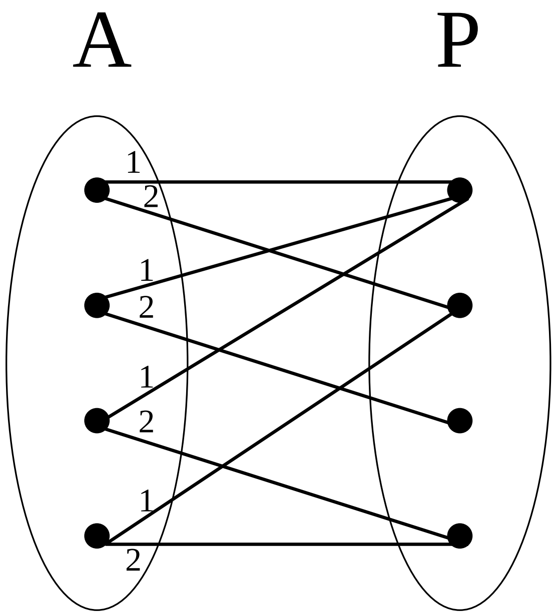
Outline of The Presentation

- Rank-Maximal Matchings
- Weighted Rank-Maximal Matchings
- Reducing Weighted Rank-Maximal Matchings to Rank-Maximal Matchings
- Popular Matchings

Rank-Maximal Matchings

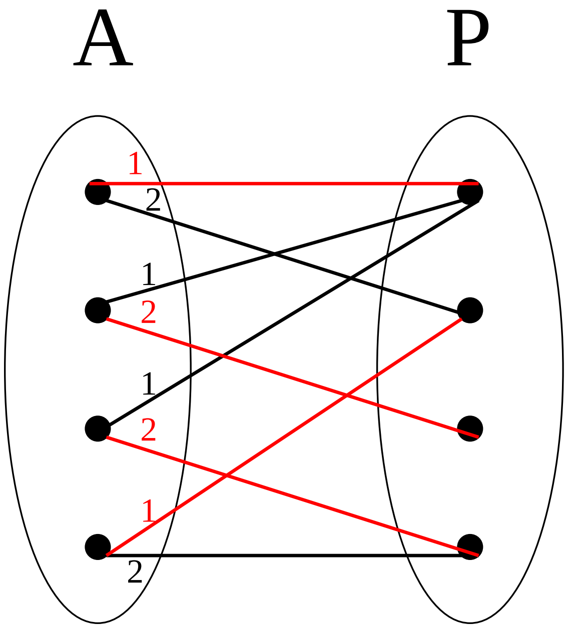
Applicants and Posts Modelled as a Bipartite Graph

- Let $\mathcal{A}$ be a set of applicants and $\mathcal{P}$ be a set of posts.
- We put an edge between an applicant a and a post p if a has a preference for post p.
- Moreover, if a has a high preference for p, we put a higher rank (lower number) for the corresponding edge.



Rank-Maximal Matchings

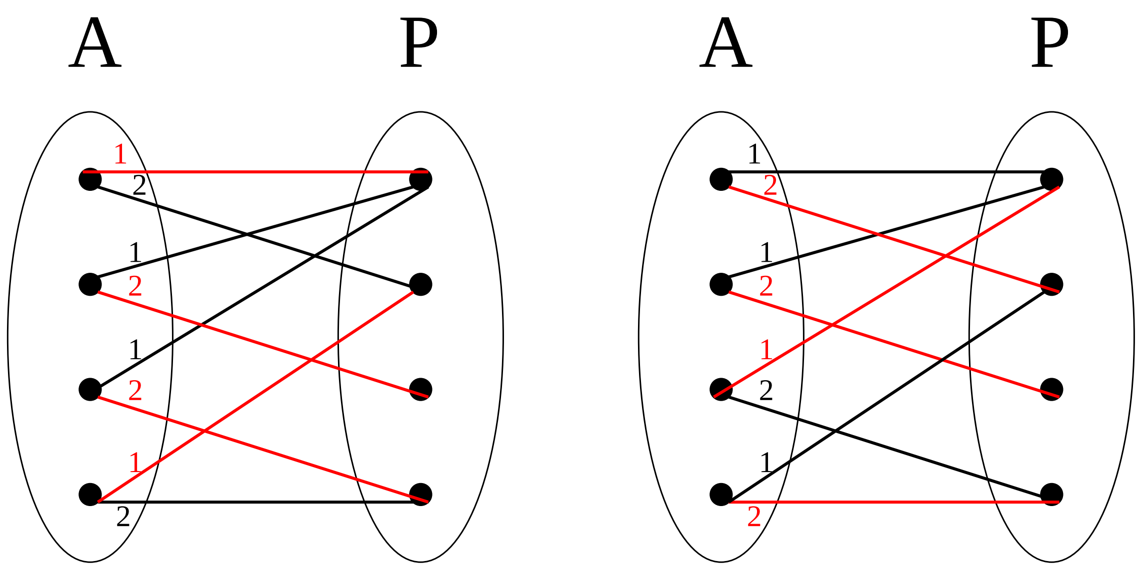
- Maximum no. of applicants are assigned their favourite posts in a rank-maximal matching.



A Rank−Maximal Matching

Rank-Maximal Matchings

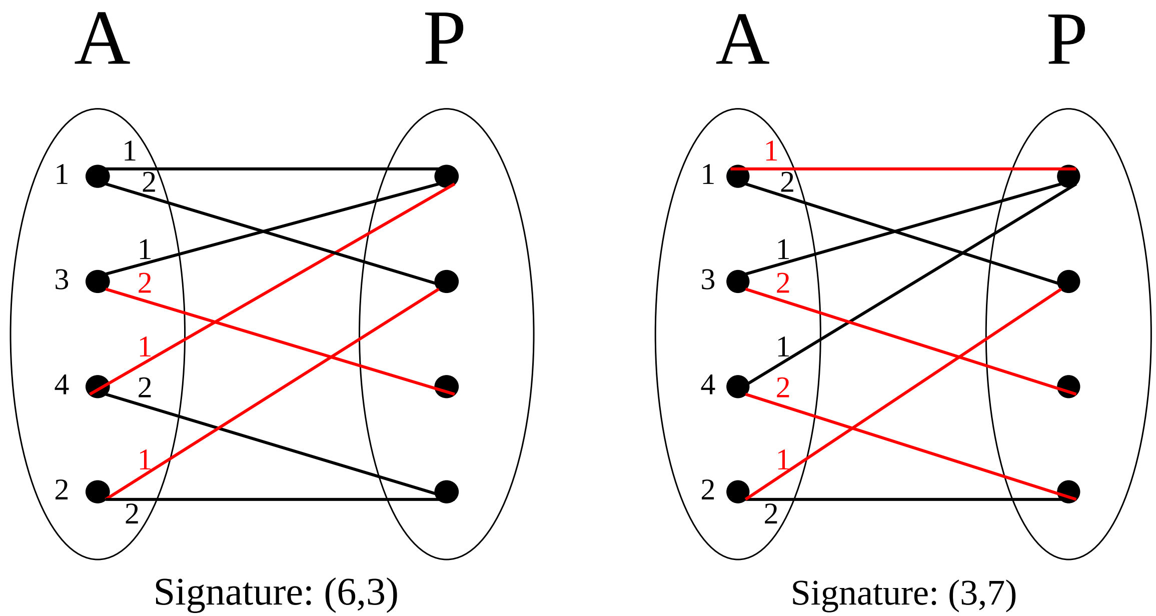
- The signature of a matching where applicants have upto r ranks is $(x_1, x_2, \cdots, x_r)$, where x_i is the no. of applicants assigned their i-th choice post.
- We want to maximize the signature lexicographically in a rank-maximal matching.

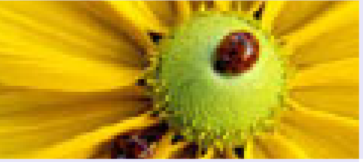


Signature: (2,2) Signature: (1,3)

Weighted Rank-Maximal Matchings

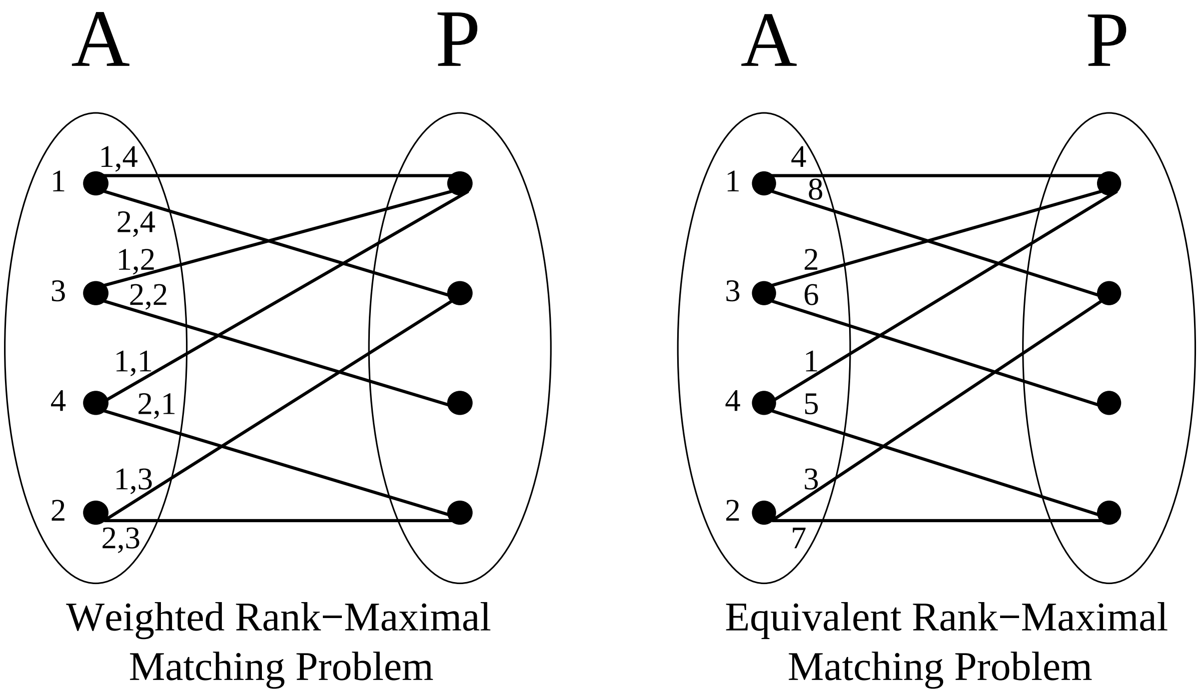
- Suppose we assign weights to each applicant.
- In the signature $(x_1, x_2, \cdots, x_r)$, x_i is now the sum of weights of applicants assigned their i-th choice post.





Reducing Weighted Rank-Maximal Matchings to Rank-Maximal Matchings

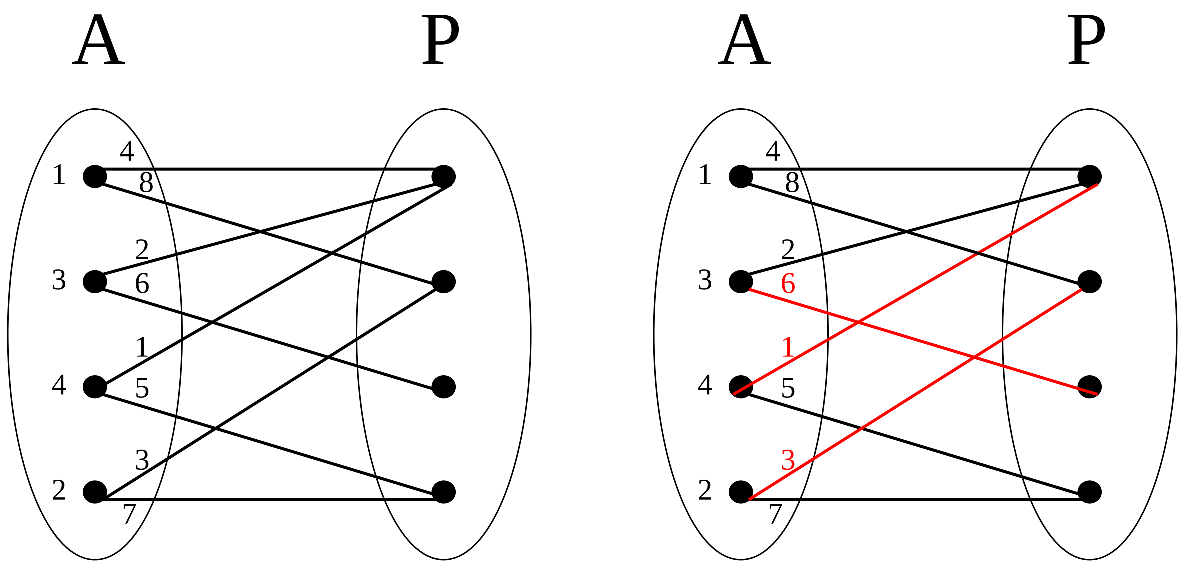
- Partition the edges E_i of each rank into k classes: $E_{i,1}, \cdots, E_{i,k}$, where $w_1 > w_2 \cdots > w_k$ are the weights and $E_{i,j}$ is the set of edges incident to applicants of weight w_j.
- Find a rank-maximal matching in $(\mathcal{A} \cup \mathcal{P}, E_{1,1} \cup E_{1,2} \cdots \cup E_{r,1} \cdots \cup E_{r,k})$.



Solving the Problem as a Problem of Rank-Maximal Matchings

Outline
Rank-Maximal Matchings
Weighted Matchings
The Problem
Reducing the Problem
The Reduction
Algorithm
Popular Matchings
Faster Algorithm

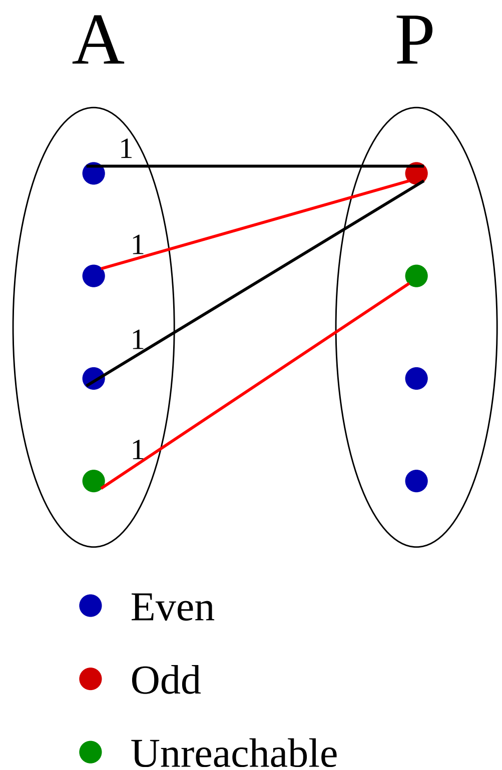
Theorem-1: The problem of finding a weighted rank-maximal matching where there are r ranks and k weights is the same as finding a rank-maximal matching in a graph with $r \cdot k$ ranks.



Signature: (1,0,1,0,0,1,0,0)

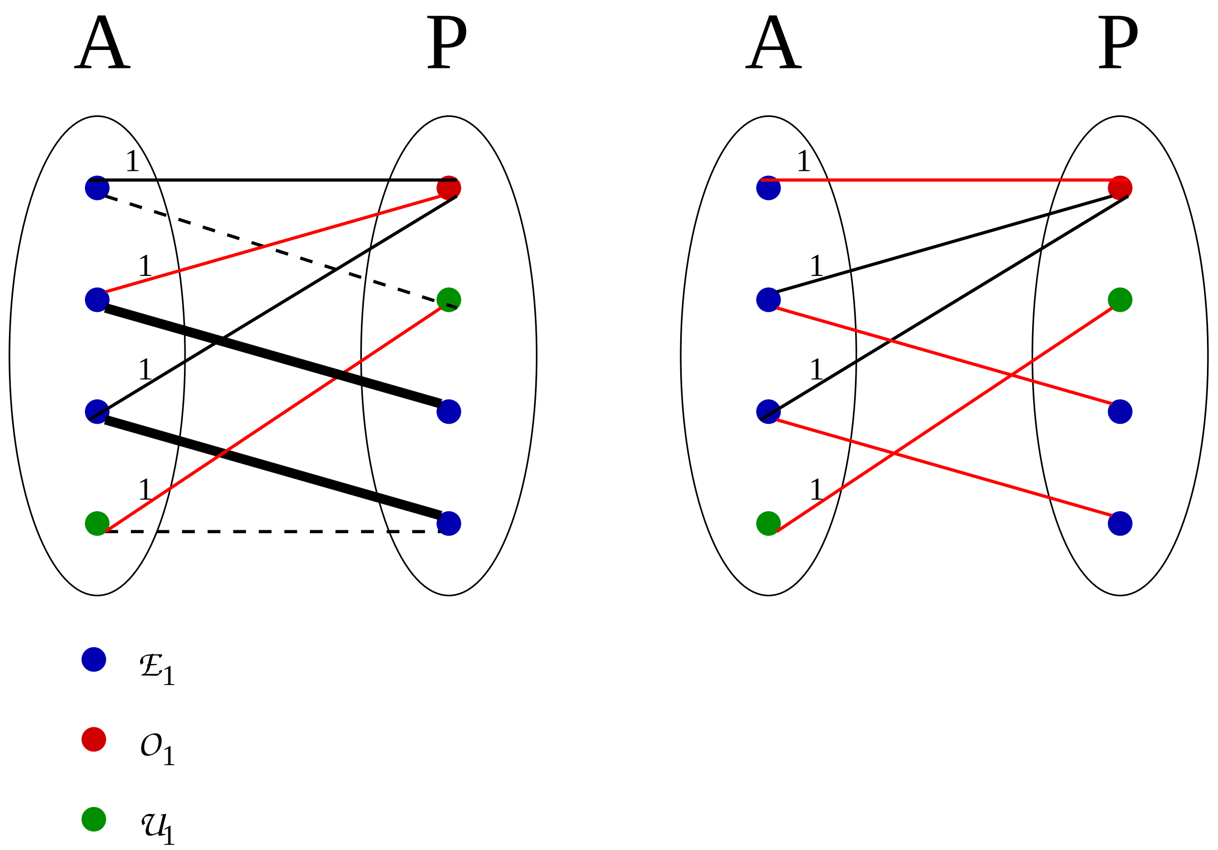
Algorithm for Rank-Maximal Matchings

- Find a maximum matching M_1 in $G_1 = (\mathcal{A} \cup \mathcal{P}, E_1)$.
- Find $\mathcal{E}_1, \mathcal{O}_1$ and $\mathcal{U}_1$, the set of even, odd and unreachable vertices in G_1.



Augmenting the Matching

- Add all edges incident only to vertices in $\mathcal{E}_1$ from E_2.
- Augment M_1 in this new graph to get a matching M_2.



Existing Algorithm for Rank-Maximal Matchings

- Add all edges incident only to vertices in $\mathcal{E}_1 \cap \cdots \cap \mathcal{E}_i$ from E_{i+1}.
- Augment M_i in this new graph to get a matching M_{i+1}.
- This approach was used by Irving et al. to give an $O(r \cdot m \cdot \sqrt{n})$ algorithm for finding rank-maximal matchings.

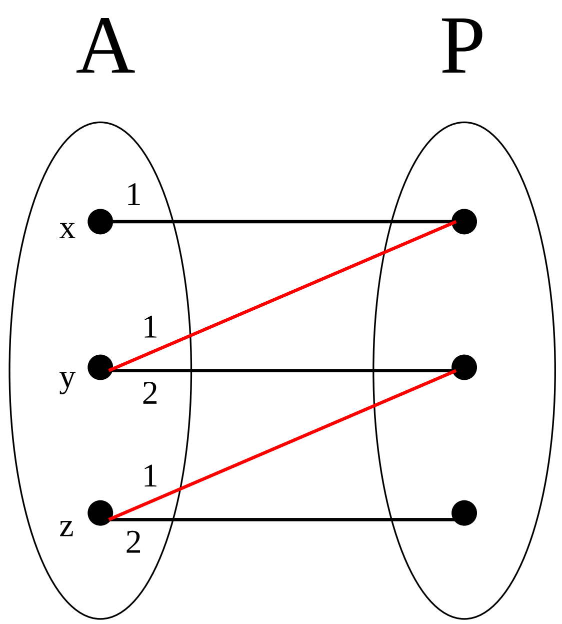
Another Approach for Finding Rank-Maximal Matchings

- The best known algorithm for finding a perfect matching in bipartite graphs [Mucha and Sankowski] runs in $O(n^{\omega})$ randomized time.
- Can we use this algorithm for finding rank-maximal matchings in $O(r \cdot n^{\omega})$ randomized time?
- Before that we will look at another problem...

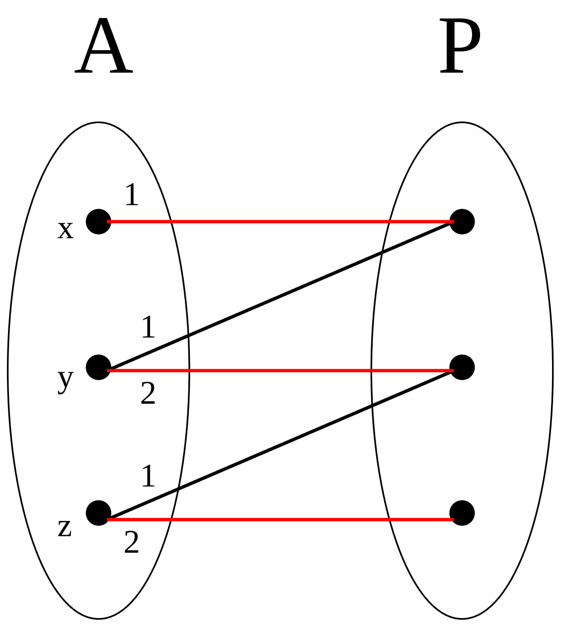
Popular Matchings

Popularity of Matchings

- A matching M' is more popular than M if the number of applicants preferring M' to M exceeds the number of applicants preferring M to M'.



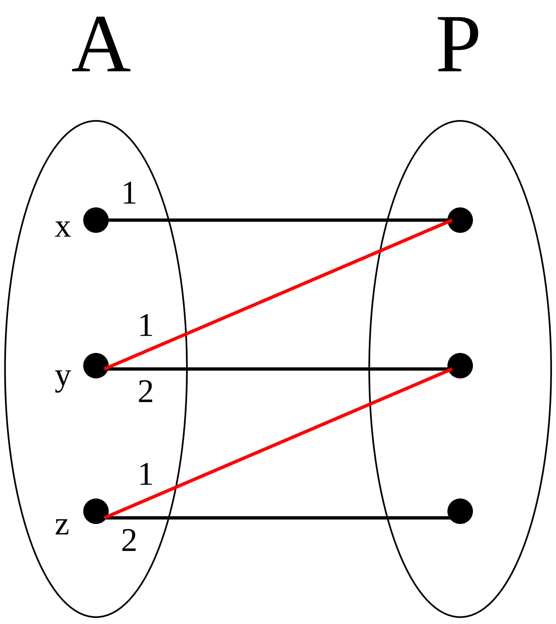
More Popular



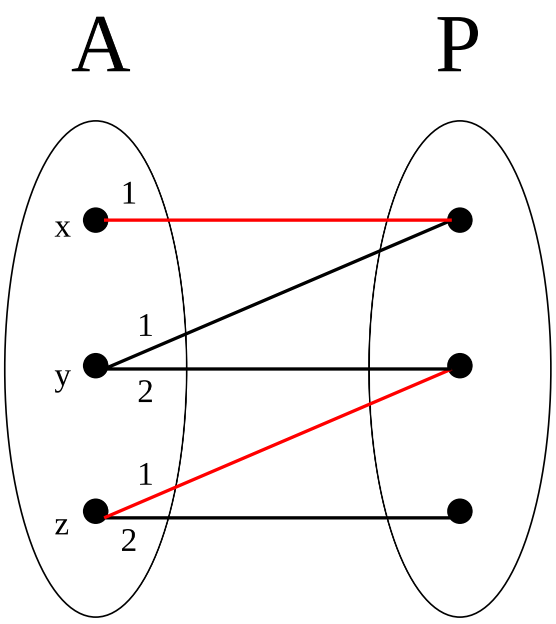
Less Popular

Popular Matchings

- A matching M is called a popular matching if there is no matching M' which is more popular than M.
- We introduce an artificial last resort post called $l(a)$ for each a so that from now on we always deal with $\mathcal{A}$-complete matchings.



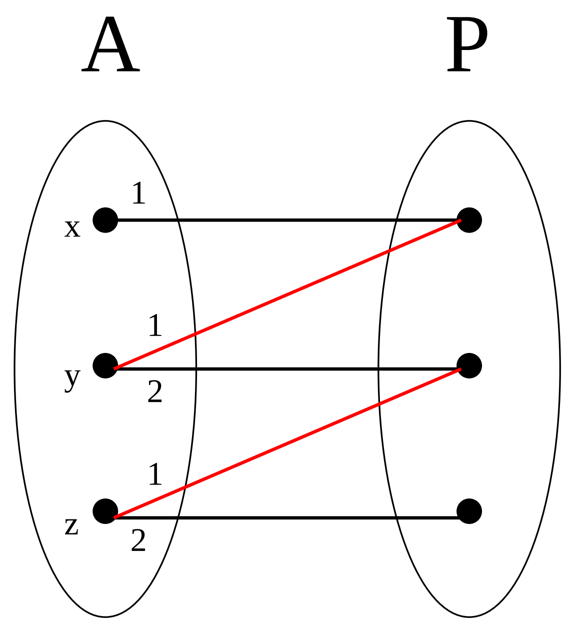
A Popular Matching



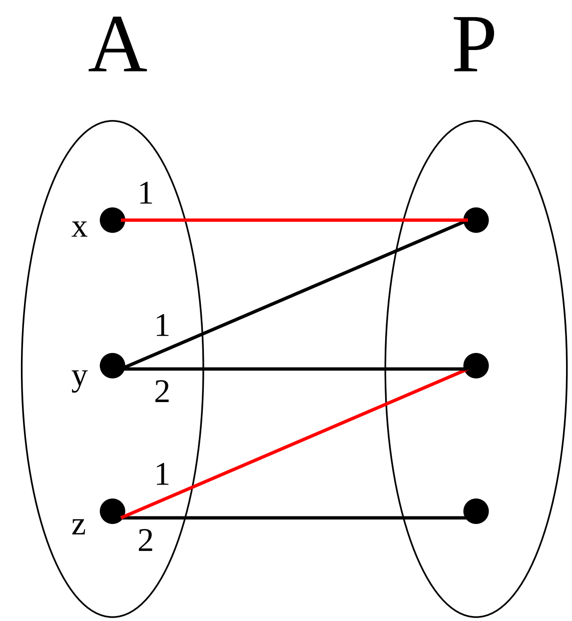
Another Popular Matching

Popular Matchings

- A matching M is called a popular matching if there is no matching M' which is more popular than M.
- We introduce an artificial last resort post called $l(a)$ for each a so that from now on we always deal with $\mathcal{A}$-complete matchings.



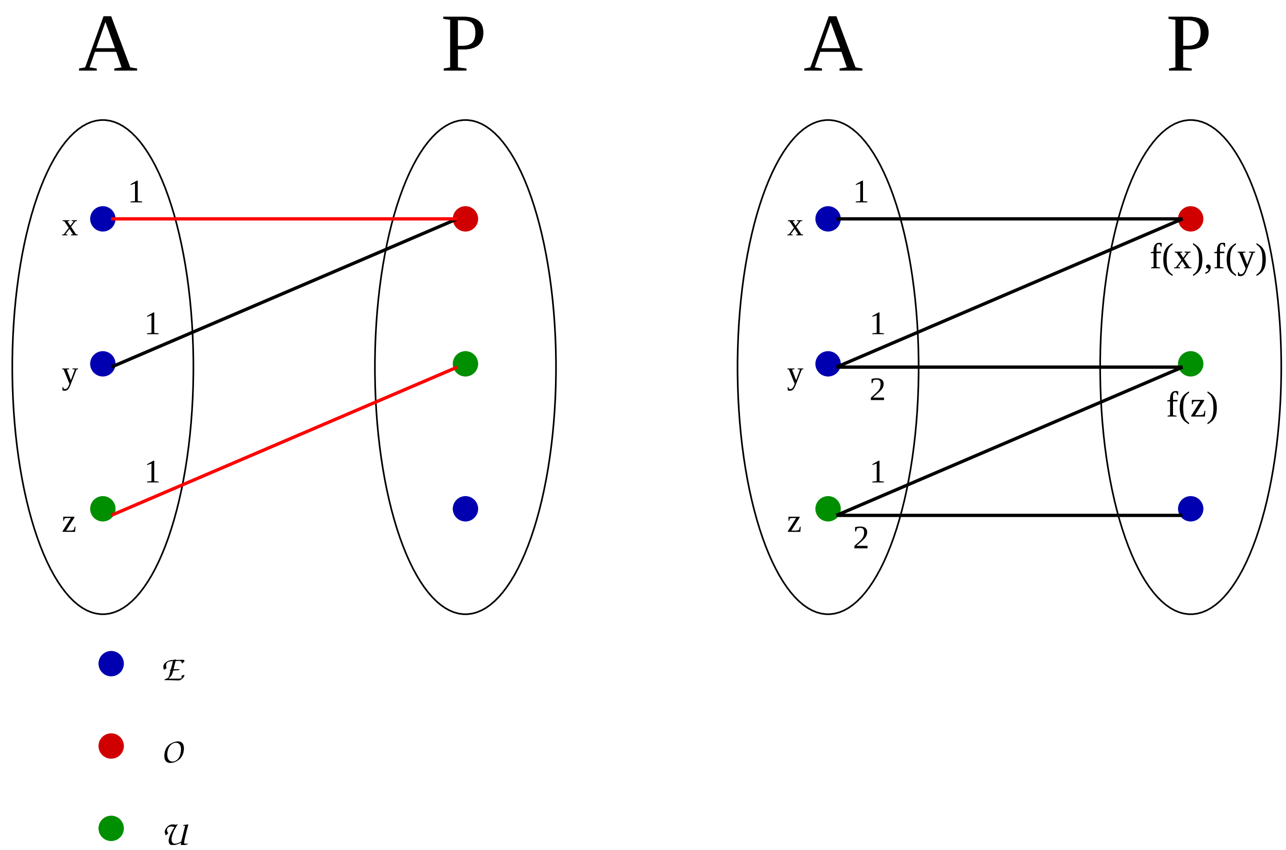
A Popular Matching



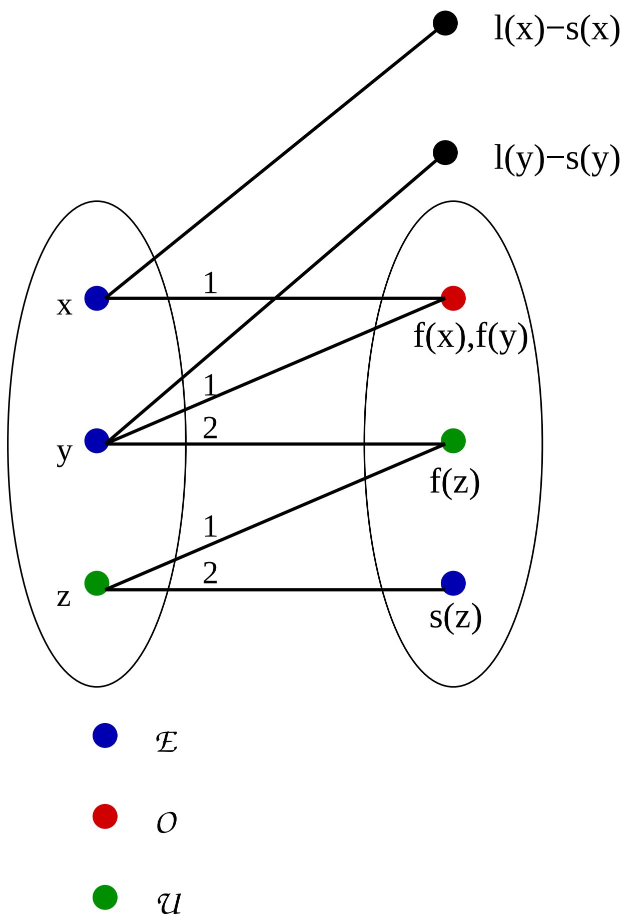
Another Popular Matching

Finding f-posts for each applicant

- In $G_1 = (\mathcal{A} \cup \mathcal{P}, E_1)$, find $\mathcal{E}, \mathcal{O}$ and $\mathcal{U}$ w.r.t M_1.
- For each applicant a find $f(a)$, the set of a's most preferred posts which belong in $\mathcal{O} \cup \mathcal{U}$.

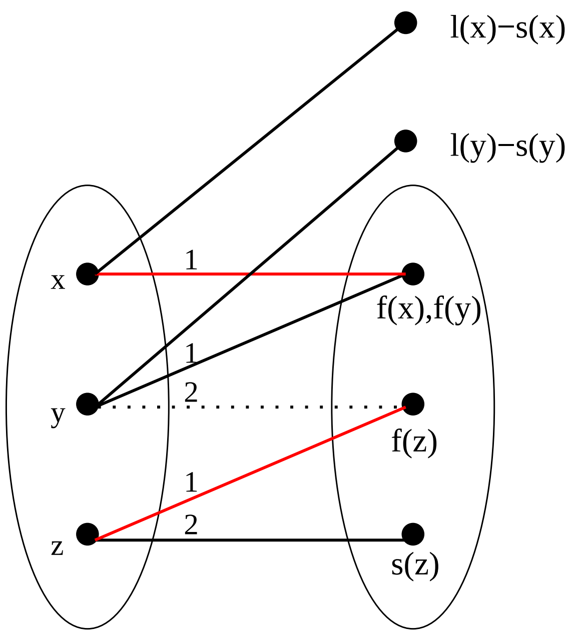


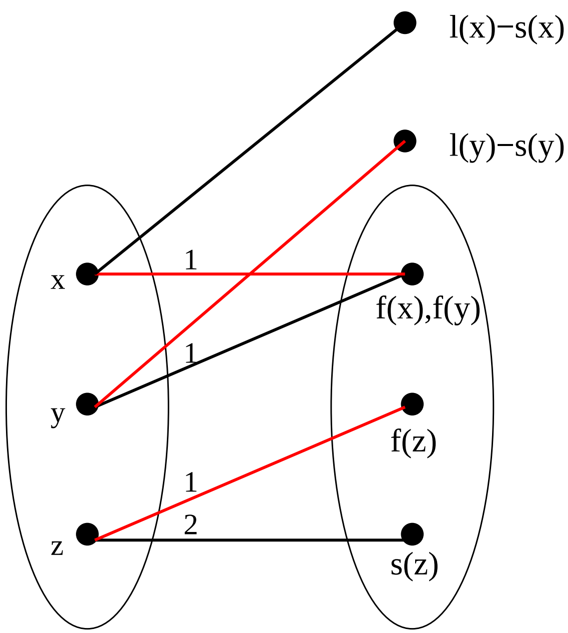
- $s(a)$ is the set of a's most preferred posts which belong in $\mathcal{E}$.
- Such a post always exists since we have introduced $l(a)$.



Algorithm for Popular Matchings

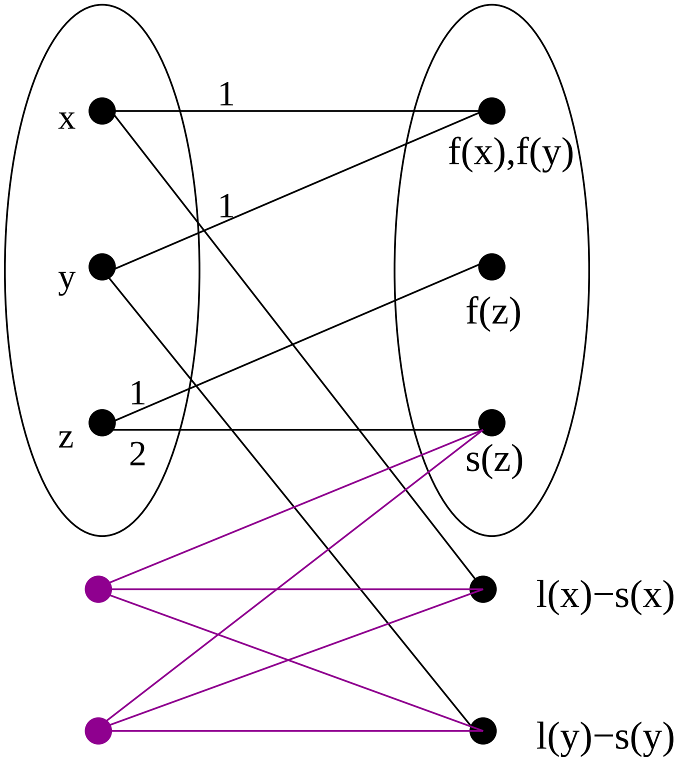
- In G, consider only edges which match an applicant a to $f(a)$ or $s(a)$.
- Start with M_1 and augment it to get a maximum matching. This is a popular matching.
- This algorithm by Abraham et al. runs in $O(m \cdot \sqrt{n})$ time.





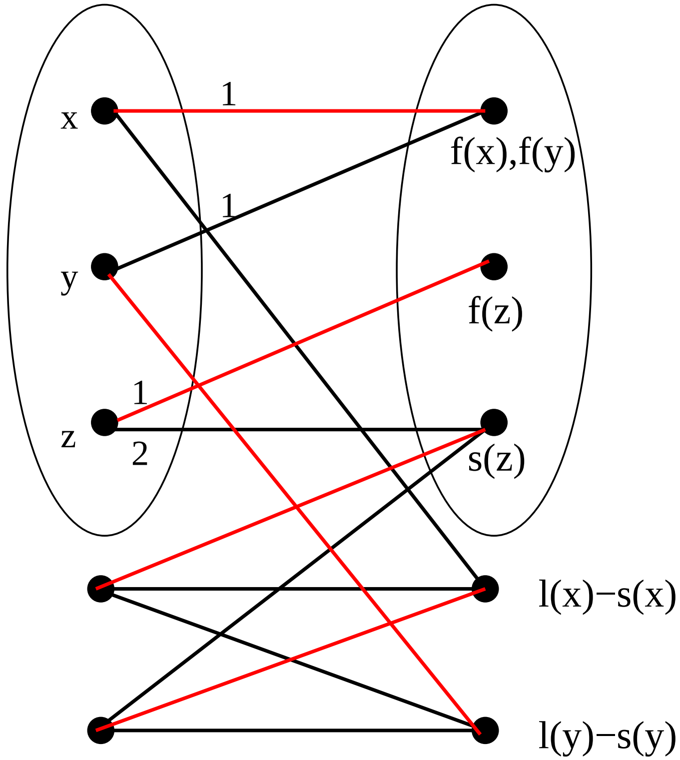
Popular Matching

- Modify G' by adding $|\mathcal{P}| - |\mathcal{A}|$ vertices to $\mathcal{A}$ and make these new vertices adjacent to all s-posts.
- Find a perfect matching in this graph using the $O(n^\omega)$ algorithm by Mucha and Sankowski.



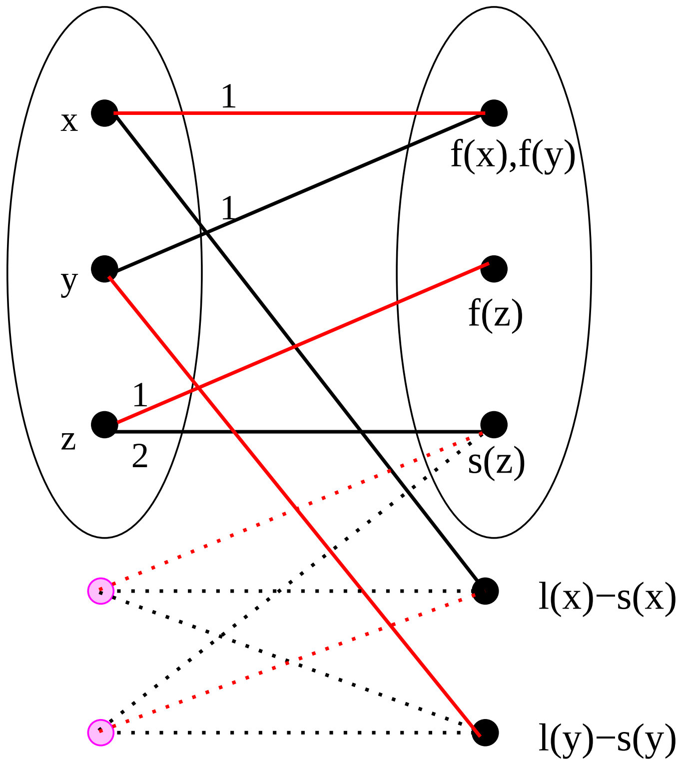
Faster Algorithm for Popular Matchings

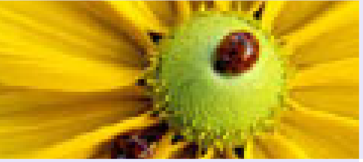
- Modify G' by adding $|\mathcal{P}| - |\mathcal{A}|$ vertices to $\mathcal{A}$ and make these new vertices adjacent to all s-posts.
- Find a perfect matching in this graph using the $O(n^\omega)$ algorithm by Mucha and Sankowski.



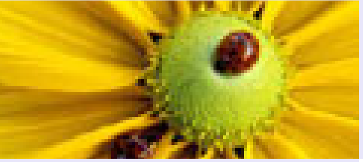
Completing the Algorithm and Finding its Running Time

- Remove the extra vertices added and edges incident upon them to get an applicant perfect matching.
- This new algorithm runs in $O(n^\omega)$ randomized time.



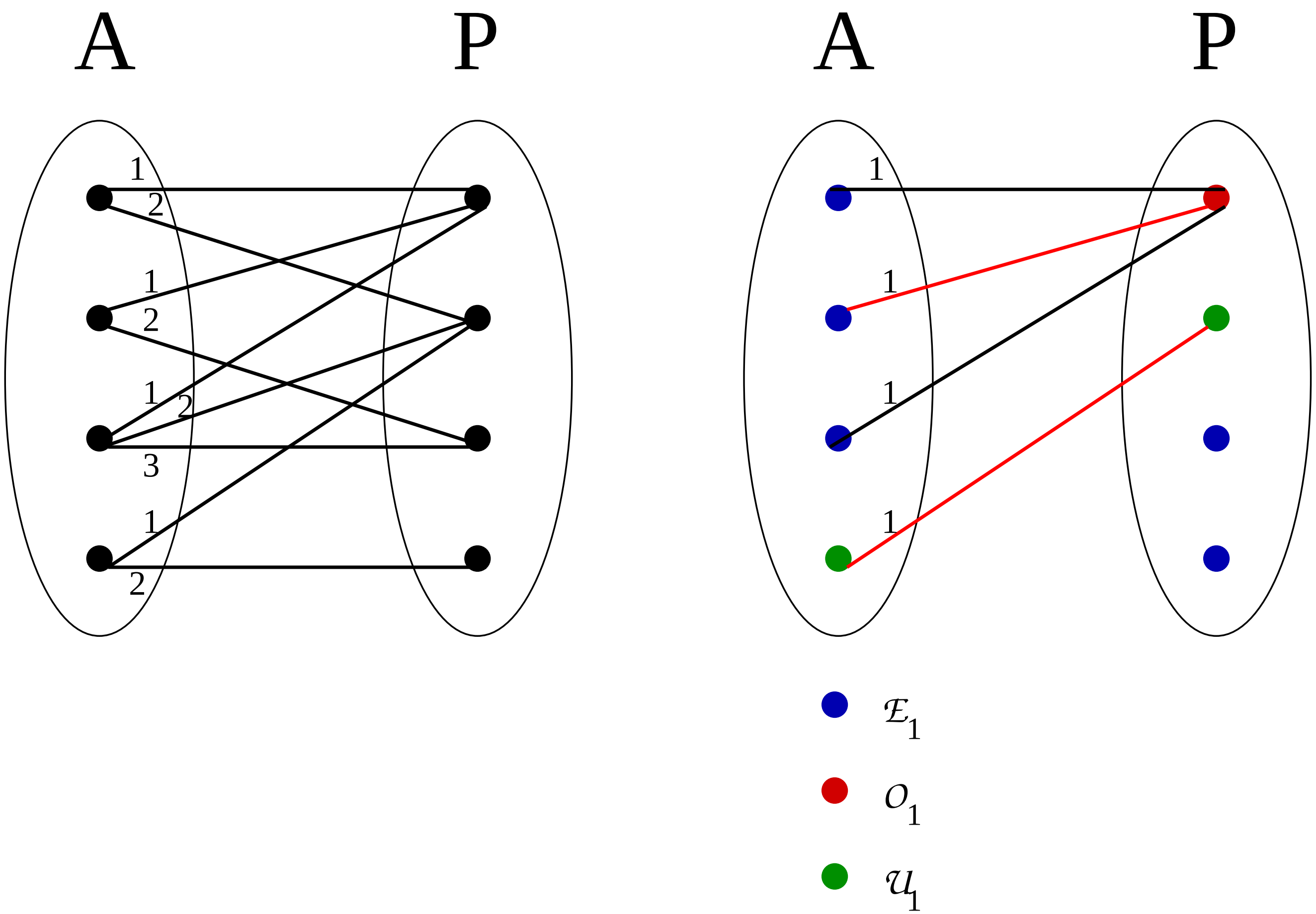


Faster Algorithm for Rank-Maximal Matchings



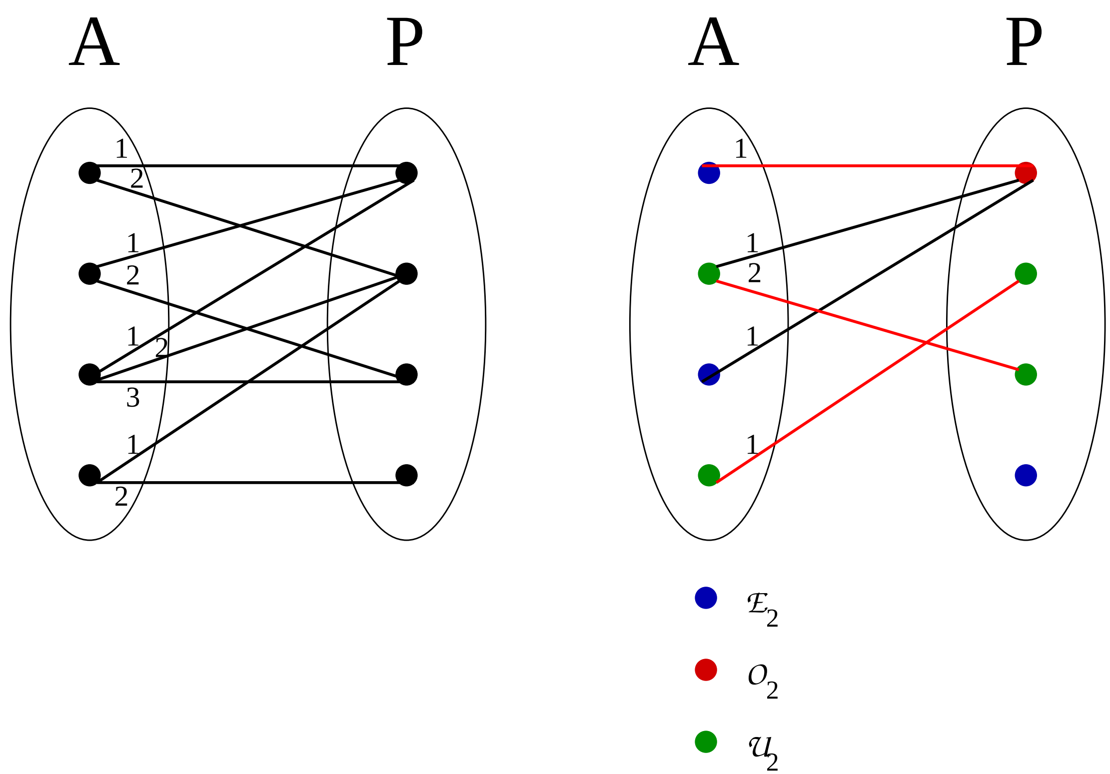
Find Even, Odd and Unreachable Vertices

- Find $\mathcal{E}_1, \mathcal{O}_1$ and $\mathcal{U}_1$ in $G_1 = (\mathcal{A} \cup \mathcal{P}, E_1)$.



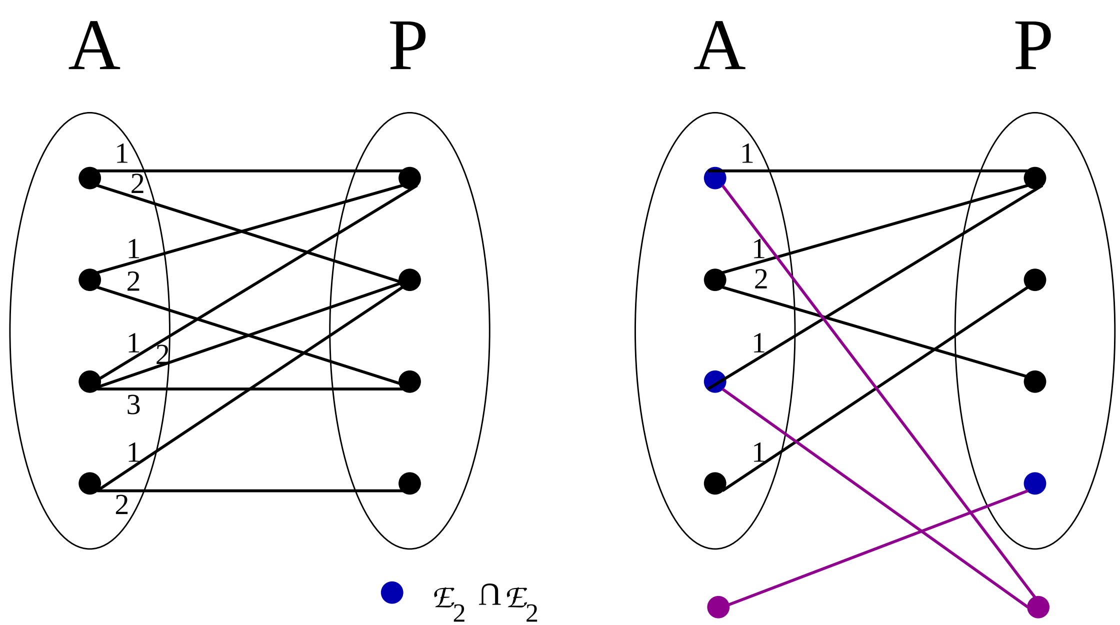
Find Even, Odd and Unreachable Vertices

- Add all edges incident only to vertices in $\mathcal{E}_1$ from E_2 to get G_2.
- Find $\mathcal{E}_2, \mathcal{O}_2$ and $\mathcal{U}_2$ in G_2.

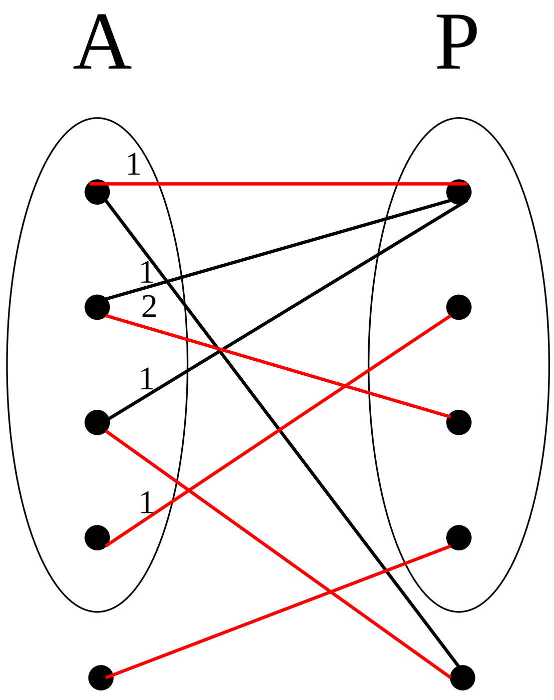


Adding New Vertices and Edges

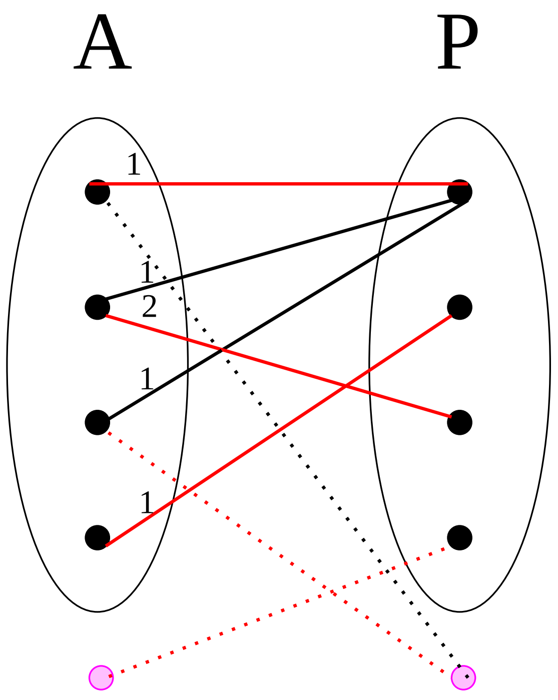
- Add all edges incident only to vertices in $\mathcal{E}_1 \cap \mathcal{E}_2$ from E_3 to get G_3.
- Add $|\mathcal{A}| - |M_2|$ vertices to $\mathcal{P}$ and make these adjacent to all vertices in $\mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{A}$.
- Add $|\mathcal{P}| - |M_2|$ vertices to $\mathcal{A}$ and make these adjacent to all vertices in $\mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{P}$.



- Find a perfect matching in this graph using the $O(n^\omega)$ algorithm by Mucha and Sankowski.
- Remove the newly added vertices and the edges incident upon them to get a rank-maximal matching.



- Find a perfect matching in this graph using the $O(n^{\omega})$ algorithm by Mucha and Sankowski.
- Remove the newly added vertices and the edges incident upon them to get a rank-maximal matching.



Conclusion

- **Theorem-2**: The Rank-Maximal matching problem can be solved in $O(r \cdot n^\omega)$ randomized time.
- The Weighted Rank-Maximal matching problem can be solved in $O(r \cdot k \cdot n^\omega)$ randomized time.